Performances des algorithmes branch-and-bound paralleles a strategie meilleur d'abord

Bernard Mans, Catherine Roucairol

UNITÉ DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tél.:(1) 39 63 55 11

Rapports de Recherche

1992

25ème anniversaire

N° 1716

Programme 1
Architectures parallèles, Bases de données,
Réseaux et Systèmes distribués

PERFORMANCE DES ALGORITHMES BRANCH AND BOUND PARALLÈLES À STRATÉGIE MEILLEUR D'ABORD

Bernard MANS
Catherine ROUCAIROL

Juin 1992

RR-1716

Performances des Algorithmes Branch-and-Bound Parallèles à Stratégie Meilleur d'Abord

---

Performances of Parallel Branch and Bound Algorithms with Best-First Search

Bernard MANS et Catherine ROUCAIROL

24 juin 1992

INRIA - Action Paradis
Domaine de Voluceau - BP 105 Rocquencourt
78153 Le Chesnay

Laboratoire MASI
Université de Versailles
45, Avenue des Etats-Unis
78000 Versailles

Résumé

Les algorithmes Branch and Bound "meilleur d'abord" sont utilisés généralement pour résoudre les problèmes d'Optimisation Combinatoire NP-difficiles. Lors de leur parallélisation, trois différents types d'anomalies sur l'accélération (*speed-up*) attendue peuvent survenir : désastreuse, d'accélération ou de préjudice d'accélération.

Un nouveau modèle de Branch and Bound parallèle, permettant d'analyser les performances parallèles espérées, est introduit.

La stratégie de recherche fondée sur la meilleure évaluation étant insuffisante pour distinguer les sommets de même évaluation lors de la sélection du nouveau candidat à la séparation, nous décrivons trois directives de sélection concernant les sommets d'évaluation identiques: la règle fifo, la règle lifo et la règle consistante.

Nous calculons pour chacune d'entres elles des bornes très fines, souvent atteignables, rendant compte des temps d'exécutions parallèles potentiels. Nous donnons des conditions nécessaires et suffisantes d'apparition de chacune des anomalies.

L'ensemble de ces résultats permet d'introduire une propriété de ***prédisposition aux anomalies***, dont nous déduisons des critères de choix précis pour les stratégies énoncées.

Différentes relaxations des hypothèses du modèle sont discutées et comparées quant à la qualité des réelles implémentations et des outils d'analyse nécessaires.

Abstract

The Branch and Bound algorithm with best first search strategy is commonly used to solve NP-hard combinatorial optimization problems. When the parallelism is introduced to increase the speed of the execution, three main anomalies on the expected speed-up may occur: detrimental, accélération and detrimental accélération.

In this paper, we design a theoretical model of the parallel best first search algorithm in order to analyse the performances.

Since the best evaluation is not always sufficient to distinguish the best node to choose with best first search strategy, three basic policies are designed for the distinct selection of same bound nodes: the fifo, the lifo and the consistent rules.

Tight bounds of each of the potential parallel executions are computed. Sufficient and necessary conditions are given regarding the predisposition for each of the three classes of anomalous behaviour. Therefore, a propriety of "proneness to anomaly", which allow several comparisons and choices, is deduced.

Further relaxations on the assumptions on the model are discussed regarding the quality on the real implementation and the analytical tools required.

1 Introduction

Les algorithmes Branch and Bound (notés B&B) sont généralement utilisés pour résoudre les problèmes d'Optimisation Combinatoire NP-difficiles[1]. Ils utilisent durant l'implémentation une file de sous-problèmes obtenus par décomposition du problème originel qu'il faut explorer. Suivant une stratégie définie *a priori*, un sous-problème (i.e. un élément de cette file) est sélectionné, puis partitionné, sauf si l'on peut prouver que le sous-problème résultant ne pourra conduire à une solution optimale ou qu'il ne pourra plus être décomposé ultérieurement.

L'utilisation du parallélisme a été développé dans le but d'améliorer les performances de la recherche durant le B&B. Lors de l'implémentation sur des multi-processeurs à mémoire partagée, la file de priorité des sous-problèmes générés reste généralement globale et est utilisée par plusieurs processeurs afin d'accélérer l'exploration des élèments de la file.

Durant les années quatre-vingt, plusieurs chercheurs ont montré que, sous certaines conditions, des implémentations parallèles de ce type ne pouvaient permettre d'obtenir les accélérations espérées.

En premier lieu, l'analyse de l'accélération (*speed-up*) calculée comme le rapport du temps de l'exécution séquentielle sur le temps de l'exécution du B&B parallèle, permet de dégager trois sortes d'anomalies :

- **anomalie d'accélération** accélération plus grande que le nombre de processeurs utilisés,
- **anomalie de décélération** accélération comprise entre un et le nombre de processeurs utilisés,
- **anomalie désastreuse** accélération inférieure à un.

En second lieu, en comparant deux exécutions parallèles, il est possible d'utiliser plus de temps avec n_2 processeurs qu'avec n_1, même si n_1 est inférieur à n_2 (**anomalie préjudiciable d'accélération** ou **préjudice de modularité de croissance**).

Les premières caractérisations de ces phénomènes sont dues à Fox et al en 1978 [3], puis Burton et al en 1983 [2]. Néanmoins, les principaux résultats sur ces phénomènes ont été obtenus par Lai et Sahni, en 1983, [10].[11]. Ils ont énoncé les conditions d'anomalies préjudiciables d'accélération, de sorte que Lai et Sprague, en 1985, [12], les ont complétées par des conditions suffisantes d'occurrence d'anomalies quand le nombre de processeurs est ou non doublé. D'une autre façon, Li et Wah, en 1984 et 1986,[14],[16],[17], s'appliquèrent à comprendre cause de l'existence d'anomalies lors de parallélisation de l'algorithme séquentiel. C'est en fait l'existence de sous-problèmes

[1] Par la suite, nous considérons toujours des problèmes de minimisation.

de même priorité lors de la sélection qui est une cause de l'apparition d'anomalies désastreuses. Si la stratégie de recherche ne peut faire un choix strict entre différents sommets de même priorité lors de la sélection, elle introduit un indéterminisme entraînant la possibilité qu'une exploration différente sera faite à chaque exécution. Ainsi, une recherche peut être mal dirigée indépendamment du type d'implémentation (séquentielle ou parallèle).

Dans une autre voie, Quinn et Deo réussirent à obtenir un majorant du speed-up atteignable par le B&B [21].

Etant donné l'avantage évident qu'il y a à préserver l'apparition d'anomalies d'accélération, des conditions interdisant par contre l'apparition d'anomalies désastreuses ont été développées. Ainsi, Li et Wah ont montré comment une gestion adéquate (*consistante*) des sommets de même priorité était suffisante pour interdire une dégradation de l'accélération. Malheureusement, une telle gestion entraîne un surcoût de calcul et de mémoire qui peut être jugé excessif alors que leur fréquence d'apparition est faible durant l'implémentation suivant une stratégie de recherche "Meilleur d'Abord". Néanmoins, leur méthode a un intérêt certain quant à la stratégie "Profondeur d'Abord" pour laquelle l'apparition d'anomalies est plus fréquente, et a donc été améliorée techniquement par Saletore et Kalé, en 1990, [23].

Le coût des anomalies devant donc être comparé au prix à payer pour interdire de tels phénomènes, il nous a paru intéressant d'étudier la conception et l'analyse de stratégies de recherche Meilleur d'Abord, prenant en compte les sommets de même priorité (évaluation), mais n'introduisant aucun surcoût de traitement ou d'espace mémoire.

Le but de cette recherche est d'étudier les conditions d'apparition d'anomalie pour trois stratégies "Meilleur d'Abord" basiques dans lesquelles l'ordre d'exploration de sommets de plus petite évaluation identique dépend de leur ordre d'arrivée : la règle **fifo** (le sommet exploré est le plus anciennement inséré dans la file), la règle **lifo** (le sommet exploré est le plus récemment inséré dans la file) et la règle **consistante** (le sommet exploré est le plus à gauche dans l'arborescence). Nous avons calculé des bornes sur le nombre d'itérations des implémentations parallèles afin de donner des critères de comparaison par rapport aux bornes obtenues pour une exploration séquentielle avec la même stratégie.

Cet article contient neuf paragraphes. Nous introduisons dans le paragraphe 2, le modèle théorique permettant d'analyser le comportement de l'algorithme de stratégie Meilleur d'Abord. Dans les paragraphes suivant (3, 4 et 5), nous étudions les bornes du nombre d'itérations et les conditions d'anomalies survenant lors de l'implémentation de chacune des trois directives de sélection introduites : la règle fifo, la règle lifo et la règle consistante. Nous comparons, dans le paragraphe 6 les résultats précédents. Nous introduisons des hypothèses d'asynchronicité dans le modèle parallèle et discutons des conséquences dans le paragraphe 7. Nous argumentons l'intérêt de nos

résultats par rapport aux autres travaux de même domaine de recherche dans le paragraphe 8, et nous concluons dans le paragraphe 9.

2 Algorithme Branch and Bound

Ce paragraphe donne les définitions et propriétés nécessaires pour concevoir et analyser des algorithmes B&B de façon théorique lors d'implémentations séquentielles et parallèles.

2.1 Algorithme Branch and Bound Séquentiel

Le but d'un algorithme Branch and Bound (noté B&B) est de résoudre un problème d'Optimisation Combinatoire, de complexité difficile (NP-complet) :

$$min\, f(x), x \in X$$

où X représente le domaine d'optimisation défini par les contraintes de cardinalité finie mais non énumérable,
x est une solution réalisable si elle appartient au domaine X,
$f(x)$ est la valeur d'une solution (fonction économique).

Le principe de l'algorithme de B&B est de décomposer un problème donné en deux ou plusieurs sous-problèmes de tailles inférieures.
Suivant une stratégie définie préalablement, un sous-problème partiel est sélectionné puis est partagé, sauf si on peut prouver que les sous-problèmes résultant ne peuvent conduire à une solution optimale, ou qu'ils ne peuvent plus être décomposés.

Afin de choisir un sous-problème parmi ceux qui n'ont pas été examinés, une évaluation (borne inférieure) des solutions dans chaque sous-problème est introduite. Un sous-problème dont l'évaluation est supérieure à la valeur de la meilleure solution connue peut être supprimé.

Plus précisément, un algorithme de B&B repose sur trois notions clés:

la séparation, principe de partition qui associe à tout problème l'ensemble des sous-problèmes en lesquels il se décompose,

l'évaluation, une fonction qui associe à tout ensemble de solutions, un nombre représentant le coût minimal de tous ses éléments,

la stratégie de parcours, une stratégie de recherche qui est utilisée pour sélectionner parmi les sous-problèmes non déjà traités, celui qui sera séparé. Le sommet sélectionné sera celui de plus petite valeur de priorité.

2.2 Arborescence du B&B

Nous donnons tout d'abord une formalisation du B&B en utilisant les notations d'Ibaraki, 1976, [6], puis nous introduisons quelques notations qui nous sont propres.

2.2.1 Formalisme d'Ibaraki

Si P_0 désigne un problème d'optimisation combinatoire, la décomposition appliquée à P_0 peut être representée par une arborescence $\mathcal{B} = (\mathcal{P}, \mathcal{E})$, où $\mathcal{P}$ est l'ensemble des sommets de $\mathcal{B}$ correspondant aux problèmes décomposés, et $\mathcal{E}$ est l'ensemble des arcs de $\mathcal{B}$ correspondant au processus de décomposition. Le problème originel P_0 est la racine de $\mathcal{B}$.

Etant donnés P_i et P_j appartenant à $\mathcal{P}$, l'arc (P_i, P_j) appartient à $\mathcal{E}$ si et seulement si P_j est généré par la décomposition de P_i.

Notations

L'ensemble des sommets dits *terminaux* de $\mathcal{B}$, désigné par $\mathcal{T}$, contient les sous-problèmes partiels qui ne peuvent plus être décomposés.
Si f désigne la fonction objectif à minimiser, $f(P_i)$ désigne la *valeur* de P_i, pour tout problème P_i appartenant à $\mathcal{T}$. Nous valuons $f(P_i)$ à l'infini si P_i est non réalisable.
Le *niveau* (ou profondeur) de P_i appartenant à $\mathcal{P}$, désigné par $l(P_i)$, est la longueur du chemin de P_0 à P_i appartenant à $\mathcal{P}$. Le niveau de P_0 est 0.
Un *ancêtre* de P_j, noté $anc(P_j)$, désigne un sommet sur le chemin de P_0 à P_j.

Définition 2.1 *L'ensemble des sommets correspondant à des solutions optimales $\mathcal{O}$ est défini par :*

$$\mathcal{O} = \{P_i \in \mathcal{P} | f(p_i) = \min\{f(p_j), , p_i \in P_i, p_j \in P_j, \forall P_j \in \mathcal{P}\}\}$$

La recherche exhaustive étant de coût prohibitif, l'algorithme B&B tend à résoudre P_0 en ne considérant qu'une petite partie de $\mathcal{P}$. Ceci est rendu possible par l'élimination des sommets P_i désignés comme ne pouvant conduire à une solution optimale pour P_0.

Définition 2.2 *Une* fonction minorante $g : \mathcal{P} \rightarrow \mathbb{R}^+ \cup \{+\infty\}$ *est calculée pour chaque sous-problème à sa création ($\mathbb{R}^+$ désignant l'ensemble des nombres réels non négatifs), et satisfait les conditions suivantes :*
(a) $g(P_i) \leq f(p) \quad \forall\, p \in P_i$
(b) $g(P_i) = f(P_i) \quad \forall\, P_i \in \mathcal{T}$
(c) $g(P_j) \geq g(P_i)$ *si P_j est un fils de P_i, $P_i \in \mathcal{P}$.*

Définition 2.3 *Si un problème décomposé P_i, P_i appartenant à $\mathcal{T}$, a une solution avec la meilleure valeur de la fonction objectif trouvée jusqu'à présent, alors la solution devient le* titulaire *de valeur z (la meilleure solution connue).*

Définition 2.4 *Un sommet de l'arborescence est dit exploré si il a été séparé en plusieurs successeurs et si chacun de ses successeurs a été évalué*

Lemme 2.1 *Chaque sommet P_j, $P_j \in \mathcal{P}$, peut être éliminé des explorations à venir si*

$$g(P_i) \geq z$$

Preuve Suivant les propriétés de la fonction de minimisation (2.2), ce sous-problème ne peut conduire à une solution de coût moindre pour P_0, et peut donc être détruit. Néanmoins, cette régle d'élimination ne s'applique qu'à la recherche d'une solution unique. Pour la recherche de l'ensemble des solutions optimales, on montrera facilement que la règle doit alors s'énoncer avec $g(P_i) > z$. □

2.2.2 Caractérisation de l'arborescence

La valeur optimale, désigné par z^*, la *fonction minorante* g et la fonction économique f nous permettent de définir quatre sous-ensembles disjoints de $\mathcal{P}$ (figure 1) :

$$\begin{aligned}
&(\textit{Arborescence Critique}) && \mathcal{C} = \{P_i \mid g(P_i) < z^*\} \\
&(\textit{Sommets d'Indecision}) && \mathcal{M} = \{P_i \mid g(P_i) = z^* \textit{ et } P_i \textit{ non terminal}\} \\
&(\textit{Sommets Optimaux}) && \mathcal{O} = \{P_i \mid g(P_i) = z^* \textit{ et } P_i \textit{ terminal}\} \\
&(\textit{Sommets Elagable}) && \mathcal{D} = \{P_i \mid g(P_i) > z^*\}
\end{aligned}$$

avec

$$\mathcal{P} = \mathcal{C} \cup \mathcal{M} \cup \mathcal{O} \cup \mathcal{D} \textit{ et } \mathcal{C} \cap \mathcal{M} \cap \mathcal{O} \cap \mathcal{D} = \emptyset \qquad (1)$$

Soit $\mathcal{A}$ l'ensemble des sommets générés mais non encore explorés, il désigne les sommets "vivants" de l'exécution B&B.

Définition 2.5 *La stratégie dite "Meilleur d'Abord" S sélectionne le sommet de $\mathcal{A}$ ayant la plus petite valeur de g.*

Tous les sommets de l'arborescence critique $\mathcal{C}$ sont explorés durant une exécution si l'on veut prouver qu'il n'existe pas de meilleure solution que le titulaire. Rappelons l'avantage principal de la stratégie meilleur d'abord, sur les autres, telles les recherches en profondeur ou largeur d'abord.

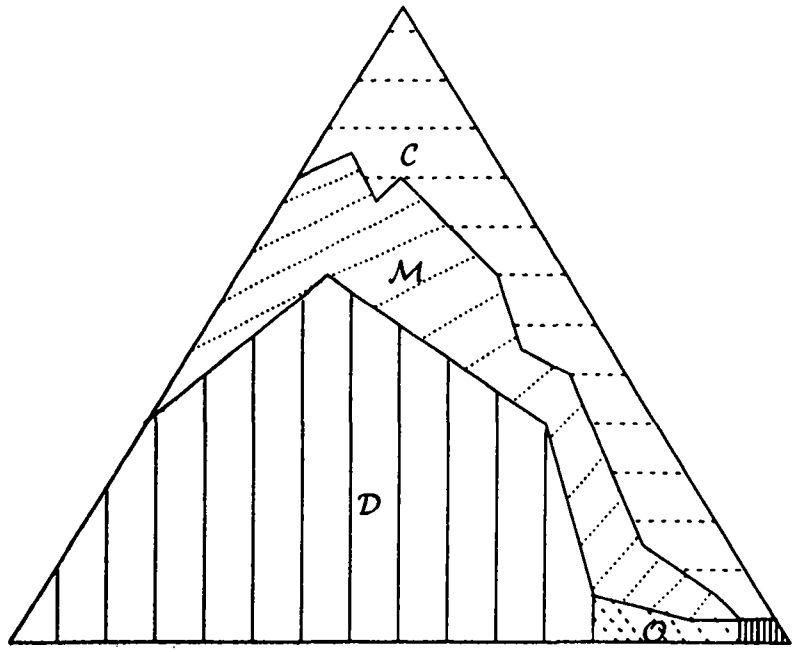

Figure 1 : Parties Disjointes de l'Arborescence de Recherche du B&B

Propriété 2.1 *L'exploration d'une arborescence de recherche suivant une stratégie meilleur d'abord interdit l'exploration de sommets de $\mathcal{D}$.*

Preuve Immédiate, Fox et al., 1978, [3], Papadimitriou et Steiglitz, 1982,[20]. □

En fait, la valeur de la borne inférieure sur les problèmes non examinés de $\mathcal{D}$ les empêche d'être explorés. Comparativement aux autres stratégies, la recherche dite "meilleur d'abord" minimise le nombre de sous-problèmes explorés.

2.2.3 Gestion des sommets de priorité égale

Aucune règle de sélection n'a été formellement introduite dans la définition de cette stratégie, lors de la présence de plusieurs sommets de même valeur d'évaluation. Un tel indéterminisme de sélection est principalement dû au fait que les implémentations utilisent généralement un tas (*implicit heap*) comme file de priorité pour les sommets vivants. Ces structures de données ne permettent pas de prédire ou prescrire un ordre de traitement des sous-problèmes de même évaluation.

Des stratégies particulières doivent donc être définies pour gérer les sommets de même évaluation.

En considérant que PlusRecent($\mathcal{A}$), PlusVieux($\mathcal{A}$) et PlusAGauche($\mathcal{A}$) désignent les ensembles de sommets de la liste de sommets vivants $\mathcal{A}$, respectivement les plus récemment générés, les moins récemment générés et les plus à gauche dans l'arborescence de recherche $\mathcal{P}$, nous définissons trois règles de gestion des sommets d'égale évaluation :

$$\textbf{(lifo)}\ S_l(\mathcal{A}) = \{P_i | g(P_i) = \min_{\forall P_j \in \mathcal{A}} g(P_j)\} \cap \{P_i | P_i \in PlusRecent(\mathcal{A})\} \quad (2)$$

$$\textbf{(fifo)}\ S_f(\mathcal{A}) = \{P_i | g(P_i) = \min_{\forall P_j \in \mathcal{A}} g(P_j)\} \cap \{P_i | P_i \in PlusVieux(\mathcal{A})\} \quad (3)$$

$$\textbf{(sequence)}\ S_s(\mathcal{A}) = \{P_i | g(P_i) = \min_{\forall P_j \in \mathcal{A}} g(P_j)\} \cap \{P_i | P_i \in PlusAGauche(\mathcal{A})\} \quad (4)$$

3 Algorithme B&B parallèle

Nous introduisons un modèle d'algorithme B&B parallèle pour permettre de caractériser les propriétés et les performances du B&B parallèle.

3.1 Un modèle d'algorithme B&B parallèle

L'introduction du parallélisme dans un B&B consiste à offrir à l'ensemble des processeurs la possibilité de sélectionner en parallèle des sommets à explorer.

Chaque processeur exécute le cycle de décomposition de la même façon qu'en séquentiel : il sélectionne le sommet de plus petite évaluation, l'explore et insére dans la liste globale des sommets vivants chaque sous-problème généré pouvant conduire à une meilleure solution. Afin de prévenir toute incohérence des structures de données partagées, une attention particulière doit être apportée à la synchronisation des différents processeurs lors de la modification du titulaire ou de la file de priorité. Un autre changement important est introduit lors de la terminaison de l'algorithme qui a lieu lorsque la file de sommets est vide et que tous les processeurs sont passifs.

Néanmoins, si l'implémentation d'un tel modèle parallèle se revèle simple, il n'en va pas de même pour l'analyse théorique. Dans la plupart des travaux relatifs à ce sujet, [11, 12, 16], l'analyse théorique de l'algorithme nécessite quatre hypothèses supplémentaires.

(A1) Filiation Directe : la fonction d'évaluation et le principe de séparation appliqués au sous-problème P_i dépendent uniquement de l'information obtenue le long du chemin entre le sommet initial racine P_0, et ce sommet P_i.

(A2) Synchronicité : au même instant, chaque processeur sélectionne un problème différent à explorer. Tous les processeurs insèrent les problèmes générés à la fin de leur étape d'exploration, tous ensemble.

(A3) Granularité Constante : la taille de travail de chaque processeur pour chaque itération de l'algorithme est constante tout le long de l'exécution et est indépendante du contexte du problème sur lequel travaille le processeur.

(A4) Aucun surcoût d'implémentation : le temps d'accès à la file des sommets vivants ou à une quelconque ressource partagée est considérée comme constante ou négligeable.

La première hypothèse est usuellement tacite car elle appartient aux règles de l'implémentation du B&B. La principale propriété d'une telle hypothèse est de garantir le même contexte au sous-problème (évaluation, degré de séparation,...) quelle que soit la stratégie choisie.

Les trois hypothèses suivantes sont fortement corrélées. Elles introduisent le concept d'*itération* d'un algorithme de B&B .

Définition 3.1 *Durant une* itération, *chacun des processeurs exécute le cycle de sélection-exploration-insertions. Dans le cas où le nombre de sommets vivants est inférieur au nombre de processeurs, les processeurs "inoccupés" attendront jusqu'à la prochaine itération.*

Propriété 3.1 *Une* itération *définit l'unité de temps de l'exécution, indépendamment de la machine ciblée pour l'implémentation.*

Ainsi, durant une itération du modèle séquentiel, un unique sous-problème est exploré. Avec un modèle à p processeurs, au plus p sous-problèmes de plus petite évaluation seront décomposés et explorés durant une itération.

Néanmoins, malgré les fortes contraintes de ces hypothèses, un complet indéterminisme existe : on ne peut déterminer ni par quels processeurs seront sélectionnés les sommets, ni dans quel ordre les sommets générés seront insérés à la fin de l'itération.

3.2 Propriétés Caractéristiques

Nous décrivons d'abord les propriétés classiques du modèle, et en introduisons quelques autres nécessaires à la détermination des bornes du nombre d'itérations.

3.2.1 B&B séquentiel optimal

La complexité des algorithmes B&B étant fortement liée au problème combinatoire résolu, de nombreux outils ont été introduits pour permettre des comparaisons entres différentes stratégies. Bien entendu, le calcul du nombre de sous-problèmes explorés est une des méthodes d'estimation de la qualité de la stratégie. Malgré cela, quelques chercheurs, tel Ibaraki, 1976, [6], ont montré l'intérêt de prendre également en compte le nombre de sommets générés durant l'exécution (si certaines hypothèses sont relachées, notamment sur le surcoût d'implémentation (A4)).

La définition suivante est communément acceptée, Fox et al., 1978,[3].

Définition 3.2 *Un algorithme B&B séquentiel qui lors d'une exécution "minimise le nombre de sous-problèmes explorés", i.e. le nombre d'itérations, est* **optimal**.

Néanmoins, si le nombre d'itérations et le nombre de sous-problèmes explorés sont identiques dans le cas d'une analyse séquentielle, cette technique de comptage ne peut être valable en parallèle, malgré les quatres hypothèses introduites. Le nombre d'itérations développées en parallèle peut être trés différent du rapport du nombre d'itérations en séquentiel sur le nombre de processeurs.

Une première raison est un nombre peut-être insuffisant de sommets vivants à chaque itération, qui ne nous garantit pas l'exploration de p sommets à chaque itération avec p processeurs. La seconde, comme déjà vu, est que des valeurs identiques de la fonction d'évaluation g peuvent entraîner des explorations différentes de l'arborescence de recherche et peuvent introduire des conflits de choix dépendant de la règle de stratégie meilleur d'abord utilisée. Une stratégie de choix ambiguë pour laquelle aucune règle stricte n'est définie sur l'ordre de sélection des sous-problèmes vivants est la principale raison d'un tel indéterminisme.

Nous en donnons un exemple dans la figure 2 où un carré symbolise le sommet correspondant à la solution optimale. L'exécution de l'algorithme B&B de recherche meilleur d'abord avec différentes règles sur l'arborescence montre les différents états de la file de sommets vivants lors de l'introduction du parallélisme (figure 3). Chaque ligne représente une itération (séquentielle ou parallèle) et décrit le sommet exploré, l'état de la file de sommets vivants suivant l'ordre de sélection, et la valeur du titulaire z. Seule la règle *sequence* induit une exécution parfaitement déterminée, les autres exécutions peuvent faire correspondre différents nombres d'itérations pour la même stratégie.

La règle lifo introduit une anomalie désastreuse : une exécution en parallèle peut prendre jusqu'à six itérations alors que l'exécution en séquentiel n'en prend que quatre. Dans ce cas, il faut alors plus de temps avec deux fois plus de ressources de calcul ; le contraire de ce qui était espéré. De la même

façon, une exécution suivant la règle fifo peut prendre autant d'itérations qu'en séquentiel, ce qui est également décevant.

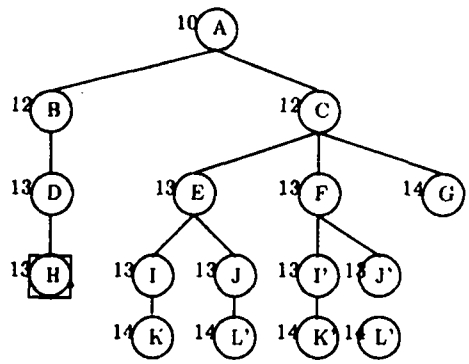

Figure 2 : Arborescence de Recherche du B&B

règle utilisée	# procs	sommet exploré	liste des nœuds vivants	valeur titulaire
fifo	1	A	BC	∞
		B	C,D	
		C	DEF,G	
		D	$\emptyset$	$13 = z^*$
lifo	1	A	CB	∞
		C	B,FE,G	
		B	DFE,G	
		D	$\emptyset$	$13 = z^*$
sequence	1	A	BC	∞
		B	C,D	
		C	DEF,G	
		D	$\emptyset$	$13 = z^*$
fifo	2	$\begin{cases} A \\ \emptyset \end{cases}$	BC	∞
		$\begin{cases} B \\ C \end{cases}$	permut(D,E,F),G	
		$\begin{cases} E \\ F \end{cases}$	D.permut(I,I',J,J'),G or $\begin{cases} D \\ E \end{cases}$ $\emptyset$	
		$\begin{cases} D \\ I \text{ or } I' \text{ or } J \text{ or } J' \end{cases}$	$\emptyset$	$13 = z^*$
lifo	2	$\begin{cases} A \\ \emptyset \end{cases}$	CB	∞
		$\begin{cases} C \\ B \end{cases}$	permut(D,F,E),G	
		$\begin{cases} F \\ E \end{cases}$	permut(I,I',J,J'),D,G or $\begin{cases} D \\ F \end{cases}$ $\emptyset$	
		(I,I',J,J')	D.permut(K,K',L,L'),G /* en deux iterations */	
		$\begin{cases} D \\ K \text{ or } K' \text{ or } L \text{ or } L' \end{cases}$	$\emptyset$	$13 = z^*$
sequence	2	$\begin{cases} A \\ \emptyset \end{cases}$	BC	∞
		$\begin{cases} B \\ C \end{cases}$	DEF,G	
		$\begin{cases} D \\ E \end{cases}$	$\emptyset$	$13 = z^*$

Figure 3 : Exemples d'Exécutions Séquentielles et Parallèles.

3.2.2 Notion de consistance

Nous introduisons maintenant quelques notations nous permettant d'analyser les anomalies.

Définition 3.3 *Sous les hypothèses (A1) à (A4), une stratégie B&B est consistante si et seulement si au moins un sommet exploré durant l'exécution en séquentiel est exploré à chaque itération de l'exécution du B&B parallèle.*

Propriété 3.2 *La règle* sequence *est consistante pour la stratégie meilleur d'abord.*

Lemme 3.1 *Soit $\mathcal{P}_{s_1}$ l'arborescence de recherche explorée avec un algorithme B&B de stratégie s_1 afin de résoudre un problème d'optimisation. Sous l'hypothèse de filiation directe (A1), un algorithme B&B de stratégie s_2 qui explore une arborescence de recherche $\mathcal{P}_{s_2}$, a trouvé la solution du problème et prouvé son optimalité si $\mathcal{P}_{s_1}$ est une sous-arborescence (au sens de l'inclusion) de $\mathcal{P}_{s_2}$.*

Preuve Soit P_i le sommet exploré générant le sommet de meilleure solution (de valeur z^*) dans l'arborescence de recherche $\mathcal{P}_{s_1}$ explorée suivant le stratégie s_1. L'exploration de la totalité de l'arborescence de recherche $\mathcal{P}_{s_1}$ par une stratégie de recherche du B&B s_2 garantit que le sommet P_i a bien été exploré, et qu'ainsi le titulaire obtenu par l'exécution suivant la stratégie s_2 est au plus de valeur z^*. Comme l'exécution de s_1 a prouvé qu'il n'existe pas de solution avec une valeur de fonction objectif inférieure à z^*, nous pouvons en déduire que le titulaire obtenu avec l'exécution suivant la strategie s_2 est égal à z^*. Ainsi, considérant l'ensemble de sommets compris dans la liste de sommets vivants durant l'exécution suivant la stratégie s_2, désigné par $\mathcal{A}_{s_2}$, lorsque le dernier sommet de $\mathcal{P}_{s_1}$ est exploré, aucun de ces sommets ne peut avoir une évaluation supérieure à z^*. Supposons que $\mathcal{A}_{s_2}$ contient un sommet P_j ayant une valeur $g(P_j)$ inférieure à z^*. Il existe un plus proche ancêtre P_k de P_j tel que $P_k \in \mathcal{P}_{s_1}$ et $P_k \in \mathcal{P}_{s_2}$ (au pire la racine P_0). Soit $P_{k'}$ le fils de P_k tel que $P_{k'} \notin \mathcal{P}_{s_1}$ et $P_{k'} \in \mathcal{P}_{s_2}$. Considérant la condition de B&B (4), $g(P_k) \leq g(P'_k) \leq g(P_j) < z^*$ ce qui contredit le fait que $P_{k'}$ n'a pu être inséré dans $\mathcal{A}_{s_1}$ ou a été détruit de $\mathcal{A}_{s_1}$. Répétant ceci, P_j n'a pu être exploré dans $\mathcal{P}_{s_1}$. Ainsi, $\mathcal{A}_{s_2}$ est vide lorsque $\mathcal{P}_{s_1}$ est totalement exploré et l'algorithme B&B avec la stratégie s_2 a prouvé l'optimalité de la solution détectée. □

Un théorème, introduit différemment par Li et Wah, en 1986, [14], se déduit immédiatement.

Théorème 3.1 *La consistance est une condition suffisante d'interdiction d'anomalies désastreuses avec la stratégie meilleur d'abord.*

Preuve La preuve est immédiate. Soit $\Phi(p)$ le nombre d'itérations nécessaires à p processeurs pour résoudre le problème. Etant donné que durant chaque itération un sommet au moins de l'arborescence de recherche obtenue séquentiellement est explorée, l'exécution du B&B parallèle nécessitera au plus $\Phi(1)$ itérations pour explorer l'arborescence de recherche obtenue séquentiellement $\mathcal{P}_1$. D'après le lemme 3.1, l'exploration globale de $\mathcal{P}_1$ terminera l'exécution de l'algorithme B&B. Ainsi, $\Phi(p)$ est inférieure à $\Phi(1)$ interdisant une accélération inférieure à un. □

3.3 Performances du B&B parallèle

Suivant le théorème (3.1), Li et Wah, 1986, [14], ont proposé d'ajouter un numéro de chemin comme deuxième clé de priorité pour chaque sommet afin de rendre consistante la stratégie meilleur d'abord. La règle *sequence* réalise une telle modification puisque le sommet sélectionné est celui de plus petite évaluation et appartient au chemin le plus à gauche. La propriété de consistance peut être obtenue par toute règle éliminant la possibilité de sommets d'évaluations identiques. Dans la règle précédente, deux sommets de même évaluation et de même numéro de chemin ne pouvant être présent au même moment dans la file de sommets vivants, leur ordre de sélection relatif est celui d'apparition le long du chemin. Ainsi, par exemple, une nouvelle règle *sequence* imposant le choix le plus à droite plutôt que le plus à gauche aurait la même propriété de consistance. Néanmoins, seule la condition d'apparition d'anomalie désastreuse est alors éliminée.

De par sa définition, la consistance est associée à la stratégie séquentielle. Son principal intérêt est d'essayer d'explorer au mieux l'arborescence de recherche séquentielle. La stratégie consistante *sequence* essaiera d'explorer le moins possible de sommets de l'arborescence de recherche séquentielle, afin de rendre possible une apparition d'*anomalie d'accélération.*

Une définition plus forte de la consistance peut être introduite :

Définition 3.4 *Sous les hypothèses (A1) à (A4), une strategie B&B est complètement consistante si et seulement si un sommet P_i n'est sélectionné avant un autre sommet P_j que lorsque son évaluation $g(P_i)$ est inférieure ou égale à $g(P_j)$*

Propriété 3.3 *Une condition nécessaire pour qu'une* anomalies d'accélération *apparaîsse est que la stratégie ne soit pas complètement consistante, [14].*

Dans ce cas, l'arborescence de recherche développée en séquentiel est trés différente de l'arborescence de recherche minimale à explorer, et l'exploration de certains sommets n'est pas nécessaire.

C'est pourquoi la notion du nombre minimal de sommets à explorer dans l'arborescence de recherche doit être introduite. Nous allons l'étudier en examinant les quatres sous-arborescences disjointes définies en (1). Nous éliminons le cas des sommets de $\mathcal{D}$ qui ne sont jamais explorés par un algorithme séquentiel de stratégie meilleur d'abord. Trivialement, un sommet terminal, P_j^* de $\mathcal{O}$, ayant la meilleure valeur de fonction objectif z^*, doit être généré. De plus, l'arborescence critique, $\mathcal{C}$, est explorée entièrement pour prouver l'inexistance de meilleure solution que le titulaire de valeur z^*. Mais, l'exploration de sommets de $\mathcal{M}$ peut être nécessaire pour atteindre P_j^*, c'est-à-dire pour le générer. Elle doit être minimisée. Le minimum de sommets de $\mathcal{M}$ sera visité si ne sont explorés que les sommets sur le plus court chemin entre deux sommets de $\mathcal{C}$ et $\mathcal{O}$.

Définition 3.5 *Dans une arborescence, la distance d'un sommet P_j à un ensemble de sommets $\mathcal{K}$, notée $d(P_j, \mathcal{K})$, est le nombre minimal de sommets sur le chemin entre P_j et un sommet de $\mathcal{K}$.*

Proposition 3.1 *Le nombre minimal d'itérations nécessaire à la résolution d'un problème par un algorithme B&B avec un seul processeur, $\Phi^*(1)$, est égale à*

$$\Phi^*(1) = |\mathcal{C}| + \min_{P_j \in \mathcal{O}} d(P_j, \mathcal{C})$$

où $|\mathcal{C}|$ est la cardinalité de $\mathcal{C}$.

De façon identique, le pire cas quant au nombre de sommets explorés pour un algorithme B&B meilleur d'abord doit être défini. Le pire des cas correspond à une exécution avec une règle de stratégie donnée qui détecte la meilleure solution le plus tard possible, de sorte que le nombre d'itérations est maximal. Le sommet P_i, qui génére le sommet P_j^* de meilleure valeur z^*, doit être le dernier de $\mathcal{M}$ a être selectionné dans la file de sommets vivants $\mathcal{A}$. Les hypothèses du B&B imposent que la totalité des sous-problèmes explorés précédemment aient une évaluation inférieure ou égale à $g(P_j^*)$. On en déduit que le pire des cas est d'explorer tous les sommets de $\mathcal{C}$ et $\mathcal{M}$ et qu'il n'existe qu'une unique solution P_j^*, dont le père est le dernier sommet exploré.

Proposition 3.2 *Avec une stratégie meilleur d'abord fixée, le nombre d'itérations d'un algorithme B&B séquentiel est borné :*

$$\Phi^*(1) \leq \Phi(1) \leq |\mathcal{C} \cup \mathcal{M}|$$

où $\Phi^*(1)$ est le nombre minimum d'itérations introduit dans la proposition (3.1).

Propriété 3.4 *Si toutes les évaluations des sous-problèmes sont différentes, la stratégie meilleur d'abord est optimale (en nombre d'itérations). Le sous-ensemble $\mathcal{M}$ est alors vide. Le nombre d'itérations $\Phi(1)$ est égale à $\Phi^*(1)$, qui est alors le nombre de sous-problèmes de $\mathcal{C}$.*

Néanmoins, l'utilisation de l'ensemble $\mathcal{M}$ dans la proposition (3.2) doit être faite avec circonspection. L'existence possible de sommets non-terminaux de même évaluation que la valeur optimale peut ne pas conduire obligatoirement à une sélection ambiguë. En effet, l'ensemble $\mathcal{M}$ peut ne pas être exploré si un sommet optimal est un fils d'un sommet de l'arborescence critique.

Proposition 3.3 *Pour que toute stratégie meilleur d'abord soit consistante, il suffit qu'un sommet optimal soit le fils d'un sommet critique (généré par son exploration).*

Preuve Soit P_{j*} le sommet de $\mathcal{C}$ qui génére un sommet P_* de $\mathcal{O}$. L'exécution séquentielle sera optimale car la génération de P_* ne nécessitera que l'unique exploration de $\mathcal{C}$. En effet, chaque sommet non critique inséré sera éliminé ultérieurement de la liste de sommets vivants. Relativement à la stratégie meilleur d'abord, chaque sommet exploré précédemment à P_{j*} appartient à $\mathcal{C}$. Dans ce cas, soit $\mathcal{P}_s$ l'arborescence de recherche explorée en séquentielle avec une stratégie meilleur d'abord, $\mathcal{P}_s$ est identique à $\mathcal{C}$. Lors de l'implémentation parallèle, supposons qu'il existe une itération E_i qui n'explore pas un sommet de $\mathcal{P}_s$. Par définition, aucun sommet de E_i appartient à $\mathcal{C}$, ils appartiennent donc tous à $\mathcal{M}$. Donc, suivant la stratégie meilleur d'abord définie, tous les sommets de $\mathcal{C}$ ont été explorés. Suivant le lemme 3.1, l'exécution est terminée ce qui contredit l'existence de l'itération E_i. De ce fait, un sommet au moins est exploré à chaque itération parallèle. □

Lemme 3.2 *Une anomalie d'accélération ne peut se produire avec une stratégie meilleur d'abord si un sommet optimal est un fils d'un sommet critique.*

Preuve D'après la proposition 3.3, dans ce cas, les anomalies désastreuse ne sont pas possibles. Soit $\mathcal{P}_s$ l'arborescence de recherche explorée en séquentielle suivant une stratégie meilleur d'abord, $\mathcal{P}_s$ est égal à $\mathcal{C}$, et est donc optimale. Le nombre d'itérations séquentielles $\Phi(1)$ est donc optimal. Ainsi, la valeur de $\underline{\Phi}(p)$ définie comme le nombre minimal d'itérations possible en parallèle ne pourra être inférieure au nombre de sommets de $\mathcal{C}$ divisé par p, interdisant toute anomalie d'accélération. □

Définition 3.6 *L'implémentation parallèle de l'algorithme de B&B est dit optimale si et seulement si elle* **minimise le nombre maximal d'itérations exécuté par un processeur** *(minimise le temps de calcul du dernier processeur actif).*

Quelle que soit la stratégie meilleur d'abord définie, le nombre d'itérations nécessaires à l'exécution de l'algorithme B&B parallèle pour explorer l'arborescence de recherche est bornée. Nous allons dénombrer les itérations du meilleur et du pire des cas.

Proposition 3.4 *Le nombre d'itérations $\Phi(p)$ d'une exécution d'un B&B parallèle de stratégie meilleur d'abord avec p processeurs est borné par :*

$$\max(\frac{\Phi^*(1)}{p}, h_C, \min_{P_k \in \mathcal{O}} l(P_k)) \leq \Phi(p) \leq \frac{(\Phi^{worst}(1) - h_{C \cup \mathcal{M}})}{p} + h_{C \cup \mathcal{M}}$$

où
h_C est la hauteur de l'arborescence critique $\mathcal{C}$, $h_C = \max_{P_i \in C} l(P_i)$
$h_{C \cup \mathcal{M}}$ est la hauteur de l'arborescence $\mathcal{C} \cup \mathcal{M}$, $h_{C \cup \mathcal{M}} = max(h_C, h_{\mathcal{M}})$
$\Phi^*(1)$ est le nombre minimal d'itérations en séquentiel (3.1)
$\Phi^{worst}(1)$ est le nombre maximal d'itérations séquentielles (3.2),
c'est-à-dire $\Phi^{worst}(1) = |\mathcal{C} \cup \mathcal{M}|$

Preuve Un minorant est donné par le rapport du nombre minimal de sommets correspondant à $\Phi^*(1)$, sur le nombre sur le nombre de processeurs. Mais, quelques précisions sont utiles pour affiner ce minorant. Tout d'abord, même avec une infinité de processeurs, on ne peut générer immédiatement (en quelques itérations) le sommet optimal de l'arborescence. Le processus de décomposition entre la racine et le sommet optimal (même le plus proche) doit s'exécuter entièrement et niveau par niveau, et est donc intrinsèquement séquentiel. Ensuite, l'optimalité de la solution détenue par le titulaire n'est prouvée que lorsque l'arborescence critique $\mathcal{C}$ est complètement explorée, c'est-à-dire lorsque le sommet le plus profond de $\mathcal{C}$ a été atteint.

De la même façon, un majorant peut être calculé. Comme nous l'avons introduit précédemment, les sommets de $\mathcal{C}$ et $\mathcal{M}$ représentent le pire des cas possible pour l'exploration, et $h_{C \cup \mathcal{M}}$ est la hauteur de cette arborescence. Le nombre d'itérations pour explorer l'ensemble de ces sommets est donc majoré par :

$$(|\mathcal{C} \cup \mathcal{M}| - h_{C \cup \mathcal{M}})/p + h_{C \cup \mathcal{M}}.$$

Intuitivement, ce majorant peut être décomposé en deux parties. La première représente le nombre maximal d'itérations durant laquelle chaque processeur détient un sommet à explorer, et où la seconde, $h_{C \cup \mathcal{M}}$, représente le nombre maximal d'itérations nécessaires pour atteindre le sommet le plus profond lorsqu'il n'y pas de travail pour tous les processeurs. □

Nous remarquerons cependant que ces bornes peuvent ne pas être atteintes. Par la suite, nous désignerons le minorant et le majorant de $\Phi(p)$ par $\underline{\Phi}(p)$ et $\overline{\Phi}(p)$, respectivement.

Définition 3.7 *L'arborescence de recherche optimale avec p processeurs est l'arborescence obtenue par l'implémentation optimale de l'algorithme de B&B avec p processeurs.*

Remarquons que suite à la possible famine de certains processeurs en parallèle, l'exploration de l'arborescence optimale en séquentiel peut ne pas conduire à une exploration optimale en parallèle avec p processeurs. L'arborescence obtenue par une implémentation parallèle optimale, $\mathcal{P}^*(p)$, peut ne pas être incluse dans l'arborescence obtenue lors de l'implémentation séquentielle optimale, $\mathcal{P}^*(1)$.

Un exemple est donné sur la figure 4 où $\mathcal{P}^*(3) = \{A, B, C, D, E, F, G, H, I, J\}$ et $\mathcal{P}^*(1) = \{A, B, C, D, E, F, H, I, K\}$. Dans ce cas, $\Phi^*(1)$ est de 9 itérations et $\Phi^*(3)$ est de 4 itérations.

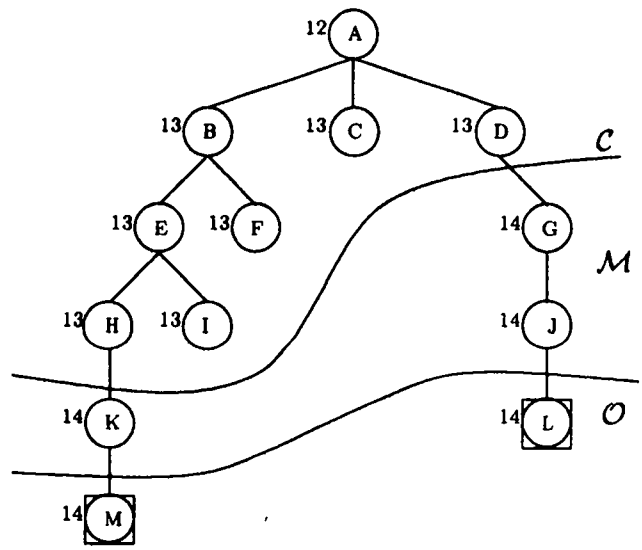

Figure 4 : Optimalité de l'Arborescence de Recherche B&B Parallèle

4 Comparaison de différentes stratégies de gestion des nœuds d'égale priorité

Après avoir introduit les performances de chacune des trois règles de gestion des nœuds d'égale priorité, nous comparons leurs différents comportements quant à l'apparition d'anomalies.

4.1 Règle fifo

Dans le cas d'une implémentation séquentielle, la règle fifo introduit un ordre de sélection des sommets dans la liste de sommets vivants et permet d'évaluer plus précisemment le nombre d'itérations.

Notation
Le sous-ensemble des sommets P_i appartenant à $\mathcal{M}$, tel que P_i a k ancêtres dans $\mathcal{M}$ est désigné par $\mathcal{M}_k$.
Le "*rang*" dans $\mathcal{M}$ de chacun des sommets de $\mathcal{M}_k$ est défini par la valeur de l'indice, k.

Lemme 4.1 *Durant une exécution séquentielle suivant une stratégie meilleur d'abord avec règle fifo, l'ensemble des sommets de $\mathcal{M}_{i-1}$ doit être exploré avant que ne le soient ceux de l'exploration de sommets de $\mathcal{M}_i$, i strictement positif.*

Preuve Par induction. Durant l'exécution séquentielle, l'exploration d'un sommet de $\mathcal{M}_0$ est possible si et seulement si tous les sommets de $\mathcal{C}$ ont été explorés. De plus, à la fin de l'exploration du dernier sommet de $\mathcal{C}$, tous les sommets de $\mathcal{M}_0$ sont dans la file $\mathcal{A}$ de sommets vivants : aucun n'a encore été exploré et aucun ne pourra être inséré ultérieurement. Puisque son père appartient à $\mathcal{M}_0$, un sommet P_1 de $\mathcal{M}_1$ sera inséré dans $\mathcal{A}$, avec une priorité inférieure à tout sommet P_0 de $\mathcal{M}_0$ (P_0 est plus vieux que P_1). En réappliquant l'argument précédent, nous prouvons le lemme. □

L'exploration de l'arborescence peut être décrite comme l'exploration par vague de chaque rang de $\mathcal{M}$. Ce lemme 4.1 nous conduit au théorème suivant.

Théorème 4.1 *Sous l'hypothèse qu'aucun sommet optimal ne soit fils d'un sommet critique, le sommet P^* de valeur optimale trouvé durant l'exécution séquentielle suivant une stratégie meilleur d'abord avec règle fifo, appartient à l'ensemble des sommets optimaux à distance minimale de l'ensemble (cf. définition 3.5) des sommets critiques $\mathcal{C}$:*

$$P^* \in \{P_i \in \mathcal{O} \mid d(P_i,\mathcal{C}) = \min_{P_j \in \mathcal{O}} d(P_j,\mathcal{C}) \}$$

Preuve Par contradiction. Soit un sommet P_l appartenant à $\mathcal{O}$. Supposons qu'il ne soit pas à une distance minimale de $\mathcal{C}$ et qu'il ait été généré avant un sommet tel que P^*. A condition qu'il n'y ait aucune solution optimale générée par l'exploration du sommet critique, P_m, le père de P_l, et P_o, le père de P_*, appartiennent tous les deux à $\mathcal{M}$. Soient $\mathcal{M}_{k_m}$ et $\mathcal{M}_{k_o}$, les ensembles auxquels respectivement P_m et P_o appartiennent, k_m est supérieur à k_o. Suivant le lemme 4.1, P_o a dû être exploré avant P_m, ce qui contredit l'hypothèse. □

La proposition suivante établit une borne du nombre d'itérations atteinte en séquentiel.

Proposition 4.1 *Soit $\mathcal{M}_{k^*}$ l'ensemble auquel appartient le père du sommet de solution optimale. Alors le nombre d'itérations de l'exécution séquentielle du B&B $\Phi_f(1)$ est compris entre :*

$$|\mathcal{C}| + \sum_{i=0}^{k^*-1} |\mathcal{M}_i| \;\leq\; \Phi_f(1) \;\leq\; |\mathcal{C}| + \sum_{i=0}^{k^*} |\mathcal{M}_i|$$

Ceci nous permet également d'obtenir des bornes plus serrées lorsque l'on compare le nombre d'itérations d'une exécution d'un B&B parallèle avec le nombre d'itérations de l'exécution séquentielle.

Proposition 4.2 *Le nombre d'itérations $\Phi_f(p)$ d'une exécution d'un B&B parallèle de stratégie meilleur d'abord suivant la règle fifo est borné inférieurement par :*

$$\max\left(\frac{\Phi_f(1) - (m-1)(h_{\mathcal{C}} - \min_{P_i \in \mathcal{M}} l(P_i))}{p},\ \underline{\Phi}(p) \right) \leq \Phi_f(p)$$

où m est l'entier représentant le nombre de chemins dans $\mathcal{M}$.

Preuve Quelle que soit la règle associée à la stratégie meilleur d'abord, l'exploration de sommets de $\mathcal{D}$ durant l'exécution parallèle est assimilable théoriquement à une itération passive due à un manque de travail. Néanmoins, l'exécution du B&B parallèle de l'arborescence de recherche séquentielle $\mathcal{P}_s$ peut ne pas être la meilleure exploration de l'arborescence de recherche parallèle possible $\mathcal{P}_p$.

Soit le cas (figure 5a) où, durant l'exécution du B&B parallèle, une amélioration a été obtenue parce qu'un sommet P_{p*} de valeur optimale a été trouvé "plus tôt" que le sommet P_{s*} de valeur optimale généré durant l'exécution séquentielle. L'expression "plus tôt" fait référence au fait que P_{p*} apparaît avant P_{s*} dans l'exécution du B&B parallèle. De tels sommets P_{s*} et P_{p*} ont

respectivement un père P_s et P_p qui doit appartenir à $\mathcal{M}$, sinon l'exécution séquentielle est optimale (cf. lemme 3.2). Soit l'ancêtre $P_{s'}$ de P_s qui a été exploré avant le sommet P_p. L'ancêtre $P_{s'}$ de P_s (qui appartient à $\mathcal{M}_{k_{s'}}$) a été exploré après $P_{p'}$, le père de P_p, appartenant à $\mathcal{M}_{k_p-1}$. En appliquant le schéma induit par la règle fifo de nouveau, le plus vieil ancêtre P_{s0} de P_{s*} appartenant à $\mathcal{M}_0$ a dû être exploré après $P_{p''}$, l'ancêtre de P_{p*} qui appartient à $\mathcal{M}_{k_{p'}-k_{s'}}$. Ainsi, la génération du sommet terminal P_{p*} plutôt que P_{s*} a économisé au moins l'exploration des sommets du chemin entre $P_{s'}$ et P_s.

Le nombre maximal de sommets éliminés est donc borné par $k_s - k_{s'}$, où $k_{s'}$ est égale à $k_p - k_{p''}$. De plus, $k_{p''}$ est borné pour P'' par $(h_C - \min_{P_i \in \mathcal{M}} l(P_i))$, qui représente le maximum de la différence de longueur de chemins entre P_{s0} et P_{p0}. Suivant le théorème 4.1, k_p est supérieure ou égale à k_s. La valeur maximale de $k_s - k_{s'}$ égale à cette différence $k_s - (k_p - (h_C - \min_{P_i \in \mathcal{M}} l(P_i)))$ sera bornée supérieurement pour k_p égale à k_s. Ainsi, la longueur maximale du chemin de $P_{s'}$ à P_s égal à $k_s - k_{s'}$ est $h_C - \min_{P_i \in \mathcal{M}} l(P_i)$.

Mais, la génération de sommet terminal P_p peut aussi éliminer tous les sommets de $\mathcal{M}$ attendant dans la liste de sommets vivants $\mathcal{A}$ d'être décomposé comme $P_{s'}$. Le nombre maximal de tels sommets est $m - 1$, c'est-à-dire le nombre de chemins différents dans $\mathcal{M}$ à l'exception de celui auquel P_p appartient. Le nombre de sommets exploré a été réduit par $(m-1) \times (h_C - \min_{P_i \in \mathcal{M}} l(P_i))$ sous la condition de ne pas dépasser le minimum global de la recherche parallèle, $\underline{\Phi}(p)$. Et, la longueur maximale du chemin de $P_{s'}$ à P_s est le nombre maximal de vagues de taille $m - 1$ à exécuter entre le début de l'exploration du chemin de P_p et le début de l'exploration des autres chemins de $\mathcal{M}$. Dans un tel cas, il faut remarquer que les sommets le long du chemin conduisant à $P_{p''}$ ont été explorés durant l'exploration de l'ensemble des sommets critiques C.

□

$$H = h_C - \min_{P_i \in \mathcal{M}} l(P_i)$$

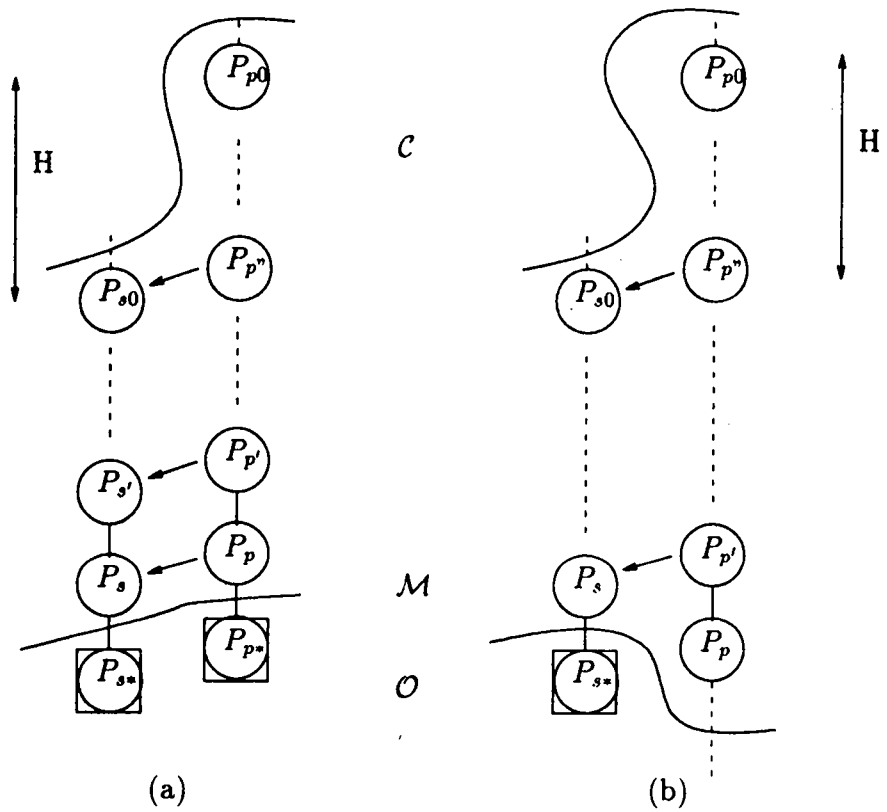

Figure 5 : Evaluation des Bornes Inf. (a) et Sup. (b) avec une Règle FIFO

Proposition 4.3 *Le nombre d'itérations $\Phi_f(p)$ d'une exécution d'un B&B parallèle de stratégie meilleur d'abord suivant la règle fifo est borné supérieurement par :*

$$\Phi_f(p) \leq \min\left(\frac{\Phi_f(1) + (m-1)(h_C - \min_{P_i \in \mathcal{M}} l(P_i)) - h_{C \cup \mathcal{M}}}{p} + h_{C \cup \mathcal{M}}, \overline{\Phi}(p)\right)$$

où m est l'entier représentant le nombre de chemins dans $\mathcal{M}$.

Preuve Dans le pire des cas pour la recherche meilleur d'abord suivant une règle fifo, l'exploration de sommets de $\mathcal{D}$ durant l'exécution d'un B&B parallèle est assimilable théoriquement à une itération passive due à un manque de travail. Par contre, l'exécution du B&B parallèle de l'arborescence de recherche avec l'ensemble des sommets de l'arborescence $\mathcal{C} \cup \mathcal{M}$, identifiera le pire cas de l'exécution et donc un majorant du nombre d'itérations possible, i.e. $(|\mathcal{C} \cup \mathcal{M}| - h_{C \cup \mathcal{M}})/p + h_{C \cup \mathcal{M}}$. La hauteur de $\mathcal{C} \cup \mathcal{M}$ a alors à être pris en compte pour les itérations où il n'y a pas assez de sommets dans la liste de sommets vivants pour occuper tous les processeurs.

Néanmoins, ce majorant peut ne pas être atteint, car l'ensemble des sommets de $\mathcal{C}$ et $\mathcal{M}$ peut ne pas avoir à être considéré entièrement. Suivant la règle fifo induisant une exploration par vague, la pire exploration par un B&B parallèle visitera l'arborescence de recherche séquentielle. Soit P_{s_*}, P_{s_*} appartenant à $\mathcal{O}$, le sommet terminal de valeur optimale généré durant l'exécution séquentielle. Soit le cas (figure 5(b)) où un sommet P_p, P_p appartenant à $\mathcal{P}_p$ est généré durant l'exécution du B&B parallèle mais n'a pas été généré durant l'exécution séquentielle, P_p n'appartenant pas à $\mathcal{P}_s$, P_p a dû être généré durant l'exécution du B&B parallèle avant P_{s_*}. Soit P_s le père de P_{s_*}, P_s appartient à $\mathcal{M}$ d'après le lemme 3.2. Soit $P_{p'}$ le père de P_p qui a été exploré avant P_s, $P_{p'}$ apppartient à $\mathcal{M}_{k_p-1}$. En appliquant le schéma d'exploration des rangs de la règle fifo, $P_{p''}$ le plus vieil ancêtre de P_p qui a été exploré avant P_{s0}, appartient à $\mathcal{M}_{k_{p'}-k_s}$. Soit P_{p0} le plus vieil ancêtre de $P_{p''}$ dans $\mathcal{M}$. Alors la longueur maximale entre $P_{p'}$ et P_{p0}, est la différence maximum entre $l(P_{s0})$ et $l(P_{p0})$, i.e.$(h_C - \min_{P_i \in \mathcal{M}} l(P_i))$. Finalement, la longueur maximum dans $\mathcal{M}$ le long du chemin de P_{p0} à P_p est égal à $k_p = (k_p - k_{p''}) + (k_{p'} - k_{p0}) = (k_s + 1) + (h_C - \min_{P_i \in \mathcal{M}} l(P_i))$. Ainsi, le nombre maximum d'exploration le long du chemin vers P_p n'appartenant pas à l'exécution séquentielle est égale à $(h_C - \min_{P_i \in \mathcal{M}} l(P_i))$.

De plus, le nombre de chemins sur lesquels un sommet n'est exploré que dans le cas d'une exécution du B&B parallèle, est borné par $m-1$, le nombre maximum de chemins dans $\mathcal{M}$ différents du chemin de P_s. Le nombre de sommets explorés durant la recherche parallèle a été augmenté par $(m-1) \times (h_C - \min_{P_i \in \mathcal{M}} l(P_i))$, sous la condition de ne pas excéder le coût maximal global de la recherche parallèle, $\overline{\Phi}(p)$.

□

Par la suite, nous désignerons par $\underline{\Phi_f(p)}$ la borne inférieure du nombre d'itérations établie par la proposition 4.2 et par $\overline{\Phi_f(p)}$ la borne supérieure du nombre d'itérations établie par la proposition 4.3.

Le résultat significatif des deux précédentes propositions est de permettre l'analyse des conditions suffisantes et nécessaires d'existence des trois types d'anomalies pouvant survenir durant les implémentations parallèles de l'algorithme B&B de stratégie meilleur d'abord suivant une règle fifo.

En premier lieu, l'accélération maximale peut être calculée et la possibilité d'anomalie d'accélération durant l'implémentation parallèle mise ainsi en évidence.

Corollaire 4.1 *La valeur de s, l'accélération pouvant être obtenue avec p processeurs, est bornée supérieurement par :*

$$s \leq p \times \frac{\Phi_f(1)}{\Phi_f(1) - (m-1)(h_C - \min_{P_i \in \mathcal{M}} l(P_i))}$$

Preuve Immédiate, découle de la proposition 4.2. L'accélération est bornée par le rapport entre le nombre d'itérations de l'exécution séquentielle $\Phi_f(1)$ et le nombre d'itérations de la meilleure exécution du B&B parallèle possible, $\underline{\Phi_f(p)}$. □

En second lieu, la condition d'existence d'anomalie désastreuse durant l'exécution du B&B parallèle donne la capacité d'amélioration effective des performances de la mise en place du parallélisme dans la résolution du problème.

Corollaire 4.2 *Une anomalie désastreuse apparaît durant l'implémentation d'une stratégie meilleur d'abord suivant une règle fifo sous la condition nécessaire suivante :*

$$(m-1) \times (h_C - \min_{P_i \in \mathcal{M}} l(P_i)) \geq (p-1) \times (\Phi_f(1) - h_{C \cup \mathcal{M}})$$

Preuve Immédiate, découle de la proposition 4.3. Une anomalie désastreuse survient si l'accélération (i.e. le rapport entre le temps de l'exécution séquentielle et le temps de l'exécution d'un B&B parallèle) est inférieure à un. Ainsi, en comparant le nombre d'itérations de l'exécution séquentielle, $\Phi_f(1)$, et le nombre d'itérations du pire cas possible d'exécution d'un B&B parallèle, $\overline{\Phi_f(p)}$, nous prouvons le corollaire. □

En troisième lieu, la modularité de croissance (extensibilité ou "scalability") de la résolution du problème par l'algorithme parallèle peut être aisément analysée en comparant sa capacité effective à utiliser un nombre croissant de processeurs.

Corollaire 4.3 *Une anomalie d'accélération préjudiciable apparaît durant l'implémentation d'un B&B parallèle de stratégie meilleur d'abord suivant la règle fifo, lorsque l'on augmente le nombre de processeurs de p_1 à p_2 processeurs ($p_2 = \rho p_1$) sous la condition nécessaire suivante :*

$$p_1 > \frac{(\rho-1)\Phi_f(1) - (\rho+1)(m-1)(h_{\mathcal{C}} - \min_{P_i \in \mathcal{M}} l(P_i)) + (h_{\mathcal{C}\cup\mathcal{M}})}{\rho(h_{\mathcal{C}\cup\mathcal{M}})}$$

Preuve Immédiate, à partir des propositions 4.2 et 4.3. Une anomalie d'accélération préjudiciable peut apparaitre lorsque l'on compare deux exécutions avec un nombre p_1 et un nombre p_2 de processeurs sachant que $p_2/p_1 > 1$, sous la condition nécessaire et suffisante que $\underline{\Phi_f(p_1)} < \overline{\Phi_f(p_2)}$, c'est-à-dire lorsque l'exécution du B&B parallèle avec p_1 processeurs peut nécessiter moins d'itérations que l'exécution du B&B parallèle avec p_2 processeurs. Ainsi, en définissant ρ, le rapport d'accroissement des ressources comme p_2/p_1, nous prouvons aisément le corollaire. □

4.2 Règle lifo

De même que pour la précédente règle, la recherche meilleur d'abord avec règle lifo permet d'avoir quelques informations quant à l'ordre de sélection des sommets de la liste de sommets vivants.

Nous introduisons d'abord plusieurs bornes serrées (c-à-d qui peuvent être atteintes) concernants l'exécution séquentielle. Il est clair que la règle lifo tend à "explorer plus en avant" dans un chemin de sommets d'évaluation égale de l'arborescence de recherche du Branch and Bound. Le nombre d'iterations est maximal lorsque les chemins visités en premier lieu dans $\mathcal{M}$, sont les plus mauvais chemins (c-à-d, ceux qui ne conduisent pas à un sommet de valeur optimale). Respectivement, ce nombre est minimal lorsque le chemin initial visité dans $\mathcal{M}$, est le meilleur chemin, c-à-d celui conduisant à un sommet de valeur optimale en passant par un minimum de sommets de $\mathcal{M}$. Dans ce cas, les bornes correspondent respectivement au pire et au meilleur cas d'exécution en séquentiel.

Proposition 4.4 *Soit $\Phi^*(1)$ le nombre optimal d'itérations en séquentiel décrit par la proposition 3.1. Alors, le nombre d'itérations de l'exécution séquentielle du B&B $\Phi_l(1)$ est compris entre :*

$$\Phi^*(1) \leq \Phi_l(1) \leq |\mathcal{C} \cup \mathcal{M}|$$

De même, dans le cas d'une implémentation parallèle appliquant cette règle, nous pouvons introduire des bornes plus fines que les bornes globales décrites dans la proposition 3.4. Les différentes bornes du nombre d'itérations d'une exécution parallèle, issues de l'analyse de situations réalisables, sont atteintes.

Proposition 4.5 *Le nombre d'itérations $\Phi_l(p)$ d'une exécution d'un B&B parallèle de stratégie meilleur d'abord suivant la règle lifo est compris entre :*

$$\max(\frac{\Phi_l(1) - |\mathcal{M}|}{p}, \underline{\Phi}(p)) \leq \Phi_l(p) \leq \min(\frac{\Phi_l(1) + |\mathcal{M}| - h_{\mathcal{C}\cup\mathcal{M}}}{p} + h_{\mathcal{C}\cup\mathcal{M}}, \overline{\Phi}(p))$$

Preuve L'exploration inutile de sommets de $\mathcal{D}$ étant assimilable théoriquement à une itération passive due à un manque de travail, la différence entre le nombre de sommets explorés en exécution séquentielle et celui d'une exécution du B&B parallèle peut atteindre au plus $|\mathcal{M}| - 1$. Clairement, ce résultat montre que les deux cas extrêmes sont possibles durant la recherche de stratégie meilleur d'abord suivant la règle lifo.

En premier lieu, l'exécution d'un B&B parallèle peut conduire à nombre d'itérations important alors que l'exécution en séquentiel a conduit quant à elle à une exploration optimale de l'arborescence de recherche du Branch and Bound. Si un chemin conduisant à un sommet de valeur optimale a été exploré en premier, l'exécution en séquentiel sera terminée sans que soit visité la part inutile de $\mathcal{M}$. Néanmoins, en parallèle, les autres processeurs coexistants peuvent générer différents sommets de $\mathcal{M}$ et les insérer aprés une insertion d'un sommet du chemin utile et avant une opération de sélection (puisque l'ordre des insertions est non déterministe). Un tel cas en parallèle conduit à une exploration complète d'un sous-ensemble de $\mathcal{M}$ qui ne contient malheureusement pas de solution réalisable, retardant le moment de travailler "plus loin dans le chemin utile". Ce cas conduit au calcul du terme de droite.

Il faut remarquer que le terme obtenu est toujours dominé par la borne supérieure de l'exécution parallèle $\overline{\Phi}(p)$ puisque $\Phi_l(1)$ est au moins égale à la cardinalité de l'arborescence critique. Néanmoins, il nous permet de décrire la différence maximale de sommets non-explorés en parallèle, c'est-à-dire de spécifier le cas d'anomalie.

En second lieu, le nombre minimal d'itérations peut être obtenu dans une exécution du B&B parallèle quand bien même l'exécution séquentielle a conduit quant à elle à une exploration importante de l'arborescence de recherche du Branch and Bound. Si le chemin conduisant au sommet de valeur optimale a été exploré en dernier, l'exécution en séquentiel se terminera par la visite de la totalité de la part inutile de $\mathcal{M}$. Néanmoins, dans le cas parallèle, les autres processeurs coexistants peuvent générer un sommet du chemin utile

de $\mathcal{M}$ et l'insérer après une insertion d'un sommet du sous-ensemble inutile et avant une opération de sélection (puisque l'ordre des insertions est non déterminisme). Un tel cas (le sommet conduisant à la solution optimale inséré en dernier), répété à chaque itération parallèle, conduit à une exploration du sous-ensemble de $\mathcal{M}$ qui ne contient malheureusement pas de solution optimale. Ce cas conduit au calcul du terme gauche de l'expression. □

Par la suite, nous désignerons par $\underline{\Phi_l(p)}$ et par $\overline{\Phi_l(p)}$, la borne inférieure et la borne supérieure du nombre d'itérations considérées dans la proposition 4.5.

Le résultat significatif que nous apportons par le biais d'une telle proposition est l'analyse de conditions suffisantes et nécessaires pour les trois types d'anomalies pouvant être obtenues durant l'implémentation parallèle de la recherche meilleur d'abord suivant la règle lifo. Nous réitérons ainsi un type d'analyse identique à celle développée pour la règle fifo et en déduisons les trois corollaires suivants.

Corollaire 4.4 *La valeur de s, l'accélération pouvant être obtenue avec p processeurs, est bornée supérieurement par :*

$$s \leq p \times \frac{\Phi_l(1)}{\Phi_l(1) - |\mathcal{M}|}$$

Preuve Immédiate, découle de la proposition 4.5. L'accélération est bornée par le rapport entre le nombre d'itérations de l'exécution séquentielle $\Phi_l(1)$ et le nombre d'itérations de la meilleure exécution du B&B parallèle possible, $\underline{\Phi_l(p)}$. □

Corollaire 4.5 *Une anomalie désastreuse apparaît durant l'implémentation d'une stratégie meilleur d'abord suivant une règle lifo sous la condition nécessaire suivante :*

$$|\mathcal{M}| \geq (p-1) \times (\Phi_l(1) - h_{C \cup \mathcal{M}})$$

Preuve Immédiate, découle de la proposition 4.5. Une anomalie désastreuse survient si l'accélération (i.e. le rapport entre le temps de l'exécution séquentielle et le temps de l'exécution d'un B&B parallèle) est inférieure à un. Ainsi, en comparant le nombre d'itérations de l'exécution séquentielle, $\Phi_l(1)$, et le nombre d'itérations du pire cas possible d'exécution d'un B&B parallèle, $\overline{\Phi_l(p)}$, nous prouvons le corollaire. □

Ensuite, le nombre minimal d'itérations peut être obtenu dans une exécution du B&B parallèle quand bien même l'exécution séquentielle a conduit quant à elle à une exploration importante de l'arborescence de recherche du Branch and Bound. Si le chemin conduisant au sommet de valeur optimale a été exploré en dernier (c'est le plus à droite), l'exécution séquentielle se terminera par la visite de la totalité de la part inutile de $\mathcal{M}$. Néanmoins, durant une exécution d'un B&B parallèle, le cas où le chemin utile est la seule partie de $\mathcal{M}$ à être exploré (durant l'exploration par les autres processeurs de la partie non-exploré de $\mathcal{C}$) peut se produire. Un tel cas conduit à ne pas explorer lors d'une exécution du B&B parallèle la partie de $\mathcal{M}$ qui ne contient pas de solution, alors que ce n'est pas le cas durant l'exécution du B&B séquentiel. □

Par la suite, nous désignerons par $\underline{\Phi_c(p)}$ et par $\overline{\Phi_c(p)}$, la borne inférieure et la borne supérieure du nombre d'itérations décrits dans la proposition 4.6.

Ces deux bornes serrées d'encadrement nous permettent de décrire sous une forme identique aux autres règles les conditions suffisantes et nécessaires d'apparition des anomalies durant l'implementation parallèle de la recherche meilleur d'abord suivant la règle consistante.

Corollaire 4.7 *La valeur de s, l'accélération pouvant être obtenue avec p processeurs, est bornée supérieurement par :*

$$s \leq p \times \frac{\Phi_c(1)}{\Phi_c(1) - |\mathcal{M}|}$$

Preuve Immédiate, découle de la proposition 4.6. L'accélération est bornée par le rapport entre le nombre d'itérations de l'exécution séquentielle $\Phi_c(1)$ et le nombre d'itérations de la meilleure exécution du B&B parallèle possible, $\underline{\Phi_c(p)}$. □

Corollaire 4.8 *Une anomalie désastreuse ne peut pas apparaître durant l'implémentation de la recherche meilleur d'abord parallèle suivant une règle consistante.*

Corollaire 4.9 *Une anomalie d'accélération préjudiciable apparaît durant l'implémentation d'un B&B parallèle de stratégie meilleur d'abord suivant une règle consistante, lorsque l'on augmente le nombre de processeurs de p_1 à p_2 processeurs ($p_2 = \rho p_1$) sous la condition nécessaire suivante que :*

$$p_1 > \frac{\rho\Phi_c(1) - \rho(|\mathcal{C}|) - (\rho+1)(|\mathcal{M}|) + (h_{\mathcal{C}\cup\mathcal{M}})}{\rho(h_{\mathcal{C}\cup\mathcal{M}})}$$

Preuve Immédiate, à partir des propositions 4.6. Une anomalie d'accélération préjudiciable peut apparaitre lorsque l'on compare deux exécutions avec un nombre p_1 et un nombre p_2 de processeurs sachant que $p_2/p_1 > 1$, sous la condition nécessaire et suffisante que $\underline{\Phi_c(p_1)} < \overline{\Phi_c(p_2)}$, c'est-à-dire lorsque l'exécution du B&B parallèle avec p_1 processeurs peut nécessiter moins d'itérations que l'exécution du B&B parallèle avec p_2 processeurs. Ainsi, en définissant ρ, le rapport d'accroissement des ressources comme p_2/p_1, nous prouvons aisément le corollaire. □

4.4 Etude Comparative

Remarquons que la comparaison des différentes stratégies meilleur d'abord n'est pas immédiate.

En effet, les résultats sur les bornes du nombre d'itérations en parallèle suivant la règle lifo peuvent également être appliqués à la recherche de stratégie meilleur d'abord suivant une règle aléatoire (aucune prédiction sur l'ordre de sélection ne peut être faite), alors que l'exploration séquentielle suivant la règle lifo d'une arborescence de recherche Branch and Bound donnée est unique et *a priori* connue.

Un tel paradoxe correspond au fait que le comportement de la stratégie suivant une règle particulière est fortement corrélé avec les données du problème pour lequel quelques exécutions particulières et extrêmes rendent impossible une analyse en moyenne. Nous soulignons le fait que certaines bornes sont atteintes pour une exécution particulière, et sont donc serrées (*tight bounds*).

Bien évidemment, les bornes atteintes permettent de comparer des stratégies de même règle (relatives à l'exécution en séquentiel suivant la même stratégie) et ainsi, ne peuvent être comparées directement. La prédisposition aux anomalies dépend de la qualité de l'exploration séquentielle. Les nombres d'itérations, $\Phi_f(1)$ et $\Phi_l(1)$, utilisées pour la description des bornes parallèles peuvent être trés différents comme le montrent les intervalles décrits pour les bornes séquentielles.

Néanmoins, si on considére plusieurs exécutions, les différences entre nombre de sommets explorés sont tous relatifs à la partie de $\mathcal{M}$ visitée ou non. La différence maximale a été donnée pour chaque règle:
$(m-1)(h_C - \min_{P_i \in \mathcal{M}} l(P_i)))$ pour la règle fifo et $|\mathcal{M}|$ pour la règle lifo.
Le cas particulier de la règle consistante entraîne une différence de $|\mathcal{M}|$ sur la borne inférieure. Mais la principale condition sur les limites (assurées par les bornes globales des exécutions parallèles, $\underline{\Phi}(p)$ et $\overline{\Phi}(p)$) est:

$$(m-1)(h_C - \min_{P_i \in \mathcal{M}} l(P_i)) \leq |\mathcal{M}| \qquad (5)$$

Ces termes représentent dans chaque cas la quantité des sommets de $\mathcal{M}$

générant le surcoût "désastreux" et le gain d'itérations en séquentiel générant l'anomalie d'accélération. Nous pouvons facilement en déduire que la différence sur les exécutions induit une différence du nombre d'itérations potentiellement plus grande pour la règle lifo.

Définition 4.1 *Une stratégie de recherche avec une règle donnée sera plus ou moins* prédisposée *aux anomalies lors de la mise en place de la parallélisation si le degré de "facilité" d'apparition de ce phénomène est plus ou moins élevé, autement dit si la taille de l'intervalle du nombre d'itérations possibles pour l'exécution du B&B est plus ou moins grande.*

Propriété 4.1 *La règle fifo est moins "prédisposée" aux anomalies que la règle lifo*

Manifestement, d'après les différents corollaires introduits, la stratégie de recherche meilleur d'abord avec la règle fifo entraîne en parallèle une possibilité d'apparition d'anomalie désastreuse plus faible que la règle lifo. Un autre avantage de la règle fifo réside dans les conditions d'anomalie d'accélération préjudiciable présentées, qui sont moins fortes que celles de la règle lifo. Mais, le désavantage majeur de cette règle fifo réside dans le fait qu'elle ne peut produire une anomalie d'accélération significative comparée à la règle lifo.

Propriété 4.2 *La règle consistante est* prédisposée *aux anomalies d'accélération mais non génératrice d'anomalies désastreuses.*

Un telle analyse confirme d'une autre manière l'avantage important de la consistance décrit par Li et Wah, [14]. Cette propriété originale, qui n'avait pas été introduite jusque là, n'est déduite que des caractéristiques du problème.

Propriété 4.3 *La règle consistante est* prédisposée *aux anomalies préjudiciables d'accélération.*

D'après les corollaires 4.6 et 4.9, la règle consistante est plus prédisposée aux anomalies préjudiciables d'accélération que la règle lifo si $\rho(\Phi_c(1) - |\mathcal{C}|)$ est inférieur à $(\rho - 1)\Phi_l(1)$, avec ρ désignant l'accroissement relatif des processeurs.

5 Vers un modèle asynchrone

Dans les applications pratiques, l'hypothèse de synchronicité (A2) et l'hypothèse de granularité constante (A3) ne sont pas vérifiées.

Le temps nécessaire au calcul d'une évaluation d'un sommet du B&B dépend du problème considéré dans l'application et change généralement tout

au long de l'exécution. Par exemple, l'évaluation du sommet le plus profond est souvent plus rapide que celle d'un de ses ancêtres proche de la racine. Le partitionnement des sous-problèmes correspond à une diminution de la taille de l'ensemble de solutions associées et est la cause majeure de ce gain en rapidité. L'implémentation asynchrone a dans ce cas de meilleures performances car elle tire profit des différentes durées de la tâche correspondant à l'exploration d'un sommet. Les différences entre granularités permettent à un processeur d'explorer un nouveau sommet dès qu'il a fini de traiter le sommet précédemment sélectionné, et ainsi éliminent l'attente d'une prochaine itération.

Malheureusement, l'analyse théorique nécessite trois fortes hypothèses, (A2) à (A4), pour le modèle parallèle. La relaxation des hypothèses de synchronicité et de granularité constante (A3) doit être ainsi envisagée et introduite dans nos résultats précédents pour rester proche du contexte des applications réelles. La définition de l'optimalité introduite dans les paragraphes précédents doit être précisée pour que notre propos reste valable.

Définition 5.1 *Lors de la résolution d'un problème donné, l'implémentation parallèle de l'algorithme B&B est optimale si et seulement si elle* **minimise la durée maximale de la tâche exécutée par un processeur**.

Il n'est pas déraisonable de supposer que la granularité d'exploration d'un sommet donné P_i d'une arborescence de recherche d'un Branch and Bound, est la même durant l'exécution en séquentiel ou en parallèle, dès que le principe de séparation et la fonction d'évaluation sont fixés.

Toutes les propriétés issues de la consistance restent vraies, lorsque l'on relâche (A2) et (A3). Si à tout instant, au moins un sommet exploré durant l'exécution séquentielle est exploré dans l'exécution du B&B parallèle, il ne peut survenir d'anomalie désastreuse.

Mais, en ce qui concerne les autres résultats, la comparaison est principalement fondée sur la différence du nombre de sommets explorés lors de différents types d'exécutions. En fait, si k_1 sommets explorés ou itérations passives ont été produits durant une exécution E_1 sans avoir été produites durant une exécution E_2, et si k_2 sommets explorés ou itérations passives ont été produites durant l'exécution E_2 sans avoir été produites durant l'exécution E_1, nous en déduisons qu'il y a une différence de coût de $|k_2 - k_1|$ explorations entre les exécutions E_2 et E_1. Néanmoins une telle analyse ne peut être correcte sans la contrainte des hypothèses (A2) et (A3), car chacun des k_2 sommets peut ne pas nécessiter le même temps pour être évalué que chacun des k_1 sommets. Nous devons donc considérer la "*qualité*" des explorations plutôt que la *quantité* de tels sommets.

Tout d'abord, la comparaison de la totalité des itérations concernant l'arborescence critique est correcte, puisque toute exécution d'un Branch and Bound doit l'explorer. Un sommet qui a été exploré durant une exécution du

B&B parallèle mais pas durant une autre exécution, doit évidemment appartenir à $\mathcal{M}$ ou à $\mathcal{D}$. Ensuite, l'hypothèse qui consiste à associer le même coût à une itération passive due au manque de travail et à une itération perdue pour l'exploration d'un sommet de $\mathcal{D}$, n'est plus possible. Un processeur qui a sélectionné un sommet inutile dans la liste de sommets vivants remplie uniquement de sommets de $\mathcal{D}$, peut finir son exploration longtemps après qu'un autre processeur ait inséré un sommet utile, de $\mathcal{C}$ ou à $\mathcal{M}$. Par contre, un processeur passif deviendra actif dès que la liste vide se remplira.

Pour répondre à cette question, Quinn et Deo, 1985 [21] ont introduit dans leur modèle un laps de temps $t(i)$ nécessaire à l'examen et la décomposition d'un sous-problème de niveau i dans l'arborescence de recherche. Mais nos expériences ont montré que cette hypothèse n'est pas acceptable (voir la troisième partie de la validation sur des problèmes connus). Les sommets de même niveau n'ont pas forcément la même granularité. En effet, les évaluations peuvent être calculées de façon différentes en ces sommets si on désire favoriser certains chemins plutôt que d'autres.

Une analyse du pire cas, où l'on sommerait les plus mauvais temps d'exploration de chaque sommet, serait évidemment injustement pessimiste, car elle ignorerait les effets corrélés des processeurs sur la gestion du remplissage de la liste de sommets vivants.

D'autre part, une analyse en moyenne concernant l'hypothèse (A3) ne peut être fidèle comme nous l'avons expliqué précédemment.

A travers ces différents points, les résultats expérimentaux connus dans la littérature, associées aux résultats théoriques, montrent qu'il serait intéressant d'étudier un modèle réaliste pour l'implémentation parallèle correspondant au monde réel (dans notre cas supportant l'asynchronisme) et ainsi de se dégager des modèles les plus rigides.

6 Discussion

Le but des précédents paragraphes est d'étudier l'impact des stratégies meilleur d'abord non-consistantes dans la parallélisation des algorithmes de Branch and Bound en l'étayant par des résultats théoriques.

Le modèle utilisé permet de prendre en compte les facteurs d'accélération dûs à l'accroisssement du nombre de processeurs et de ralentissement dûs au travail inutile effectué. L'étude est faite *a posteriori*, c'est-à-dire nécessite des informations qui ne sont obtenues qu'en fin d'exécution.

Les expériences passées dans le domaine de la résolution de problèmes d'Optimisation Combinatoire montrent que la fonction d'évaluation est fréquemment de même "qualité" pour les différentes tailles d'un problème de même type, c'est-à-dire de même proximité à la valeur optimale. De plus, la courbe de répartition des différentes valeurs des sous-problèmes de l'arbores-

cence Branch and Bound est souvent exponentielle, et fournit ainsi beaucoup de sommets non-terminaux d'évaluations égales à la valeur optimale.

De sorte qu'il est intéressant d'identifier et d'analyser les problèmes d'Optimisation Combinatoire pour lesquels nous pouvons déterminer une règle de gestion des sommets de valeur identique durant une stratégie meilleur d'abord.

Des tests sur un problème de petite taille avec quelques processeurs peuvent fournir les paramètres de réglage permettant d'améliorer les choix de la résolution d'une plus grande instance du problème sur une machine extensible.

Les autres approches données dans la littérature pour résoudre le problème des anomalies ne sont pas complètement satisfaisantes. Le surcoût introduit pour assurer la consistance dans la gestion des structures de données, [14, 23], est excessif. Les autres règles sans surcoût peuvent être avantageusement implémentées pour certaines d'entres elles, [13, 18]. Enfin, la consistance ne peut être la réponse générale aux anomalies puisqu'elle n'empêche pas l'apparition d'accélération préjudiciable.

De plus, la recherche d'une meilleure exploration de l'arborescence Branch and Bound n'est pas prise en compte par cette approche qui intervient seulement pour guider l'exploration parallèle le long du parcours séquentiel.

Pour conclure, nous analysons les différentes conséquences engendrées par le respect des hypothèses.

L'hypothèse de *filiation directe* sur la fonction d'évaluation et le principe de séparation fait généralement partie intrinsèque de l'algorithme du Branch and Bound. Comme il a été précisé dans le paragraphe précédent, les hypothèses synchrones ne peuvent facilement être levées lors des implémentations réelles. Mais l'hypothèse (A4) qui concerne le surcoût dû aux accès aux ressources partagées peut rester valable en pratique. Il faut alors s'assurer d'accès concurrents à coût constant ce qui est réalisable dans certaines structures de données spécifiques au Branch and Bound, comme nous l'avons montré dans [13, 18]. Dans le cas contraire, le surcoût d'accès croît avec le nombre de processeurs concernés, et conduit alors à l'altération des performances.

7 Conclusion

Nous avons caractérisé et analysé l'apparition d'anomalies dans des algorithmes de Branch and Bound parallèles à stratégie de recherche meilleur d'abord. La définition d'un modèle précis nous a permis d'analyser le temps d'exécution en nombre d'itérations indépendamment de la machine choisie pour l'implémentation. Des bornes du nombre d'itérations en séquentiel et en parallèle sont obtenues. Mais elles sont trop générales pour être utilisables

et fournir des informations sur les conditions nécessaires et suffisantes d'apparition d'anomalies pour la stratégie meilleur d'abord. Nous avons ainsi été amené à formuler des bornes plus précises et plus exploitables.

Nous avons montré l'importance de la gestion des sous-problèmes de même évaluation ce qui nous a conduit à définir différentes règles simples d'ordonnancement des sommets. Nous avons étudiées trois règles différentes de sélection de ces sommets: la règle fifo, la règle fifo et la règle consistante. Pour chacune d'entre elles, nous avons fourni des bornes serrées sur les performances d'exécutions des Branch and Bound. L'introduction d'une propriété de "prédisposition aux anomalies" permet de comparer les règles et de proposer des choix simples pour assurer les performances désirées lors de la parallélisation.

Nous avons montré que la règle fifo est moins susceptible de provoquer des anomalies que les deux autres règles. De plus, nous avons prouvé que la règle consistante, qui avait été développée pour faire face au problème des anomalies désastreuses, ne peut empêcher l'apparition d'anomalies d'accélération désastreuses.

Références

[1] J.E. Beasley. Supercomputers and operational research. *Journal of Operational Research Society*, 38:1085–1089, 1987.

[2] F.W. Burton, M.M. Huntbach, G.P. McKeown, and V.J. Rayward-Smith. Parallelism in branch-and-bound algorithms. Research Report CSA-3-1983, University of East Anglia, Norwich, 1983.

[3] B. Fox, J. Lenstra, A. Rinnooy Kan, and L. Schrage. Branching from the largest upper bound: folklore and fact. *Europ. J. Op. Res.*, 2:191–194, 1978.

[4] S. Huang and L.S. Davis. A tight upper bound for the speedup of the parallel best-first branch-and-bound algorithms. TR CS-TR-1852, CAR-University of Maryland, College Park, MD 20742, May 1987.

[5] T. Ibaraki. Computational efficiency of approximate branch-and-bound algorithms. *Mathematical Operational research*, 1(3):287–298, 1976.

[6] T. Ibaraki. Theoretical comparisons of search strategies in branch-and-bound algorithms. *International Journal of Computer and Informations Sciences*, 5(4):315–344, 1976.

[7] T. Ibaraki. Depth-m search in branch and bound algorithms. *Int. J. Comput. Inform. Sci.*, 7(4):315-343, 1978.

[8] M. Imai, T. Fukumura, and Y. Yoshida. A parallelized branch-and-bound algorithm implementation and efficiency. *Syst. Comput. Contr.*, 10(3):62–70, 1979.

[9] R. M. Karp and Y. Zhang. A randomized parallel branch and bound procedure. In *Proc. of the ACM Symposium on theory of computing*, pages 290–300, 1988.

[10] T.-H. Lai and S. Sahni. Anomalies in parallel branch-and-bound algorithms. In *Proc. Int. Conf. Parallel Processing*, pages 183–190, 1983.

[11] T.-H. Lai and S. Sahni. Anomalies in parallel branch-and-bound algorithms. *Communication A.C.M.*, 27:594–602, June 1984.

[12] T.-H. Lai and A. Sprague. Performance of parallel branch-and-bound algorithms. *IEEE Trans. Comput.*, C-34:962–964, 1985.

[13] B. Le Cun, B. Mans, and C. Roucairol. Opérations concurrentes et files de priorité. RR 1548, INRIA, BP105, 78153 Le Chesnay, France, November 1991.

[14] G. Li and B.W. Wah. Coping with anomalies in parallel branch-and-bound. *IEEE Trans. on Computers*, C-35(6):568–573, June 1986.

[15] G.-J. Li. *Parallel processing of combinatorial search problems.* PhD thesis, Purdue Univ., West Lafayette, IN, December 1985.

[16] G.-J. Li and B. W. Wah. Computational efficiency of parallel approximate branch-and-bound algorithms. In *Proc. 1984 Inter. Conf. on Parallel Processing*, pages 473–480, August 1984.

[17] G.-J. Li and B. W. Wah. Computational efficiency of parallel approximate branch-and-bound algorithms. Tech. Rep. TR-84-6, School Elec. Eng., Purdue Univ., West Lafayette, IN, February 1984.

[18] B. Mans and C. Roucairol. Concurrency in priority queues for branch and bound algorithms. RR 1311, INRIA, BP105, 78153 Le Chesnay, France, October 1990.

[19] R. Mehrotra and E. F. Gehringer. Superlinear speedup through randomized algorithm. In *Proc. 1985 Inter. Conf. on Parallel Processing*, pages 291–300, 1985.

[20] C.H. Papadimitriou and K. Steiglitz. *Combinatorial optimization algorithms and complexity.* Prentice Hall, 1982.

[21] M.J. Quinn and N. Deo. An upper bound for the speedup of parallel branch and bound algorithms. In *Proc. of the 3rd Conf. on Found. of Software Technology and Theoretical Computer Science*, pages 488–504, Bangalore, India, 1985.

[22] C. Roucairol. *Du séquentiel au parallèle: la recherche arborescente et son application à la programmation quadratique en variable 0-1.* Thèse d'état, Université Paris VI, June 1987.

[23] V.A. Saletore and L.V. Kalé. Consistent linear speedups to a first solution in parallel state-space search. In *AAAI90-Proceedings eighth national Conference on Artificial Intelligence*, volume 1-2, pages 227–233, July 1990.

[24] B. Wah and Y.W. Eva Ma. Manip: a multicomputer architecture for solving combinatorial extremum search problems. *IEEE Transaction on Computers*, C(33):377–390, May 1984.

[25] B. W. Wah, G.-J. Li, and C.-F. Yu. Multiprocessing of combinatorial search problems. *IEEE Computer*, pages 93–108, June 1985.

[26] B. W. Wah and C.-F. Yu. Stochastic modeling of branch-and-bound algorithms under a best-first search. *IEEE Trans. Software Eng.*, SE-11(922-934), September 1985.

ISSN 0249-6399